Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jernej Bevk

Spletna aplikacija za upravljanje študijskih sporazumov študentov na izmenjavi

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Alenka Kavčič

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Spletna aplikacija za upravljanje študijskih sporazumov študentov na izmenjavi.

V okviru diplomske naloge razvijte spletno aplikacijo, ki podpira celoten proces priprave in dopolnjevanja študijskih sporazumov za študente, ki odhajajo na mednarodno študijsko izmenjavo v okviru programa Erasmus+. Aplikacija naj bo namenjena tako študentom na študijski izmenjavi v tujini kot tudi koordinatorju za mednarodne izmenjave, kateri preko posebnih administratorskih pravic lahko upravlja tudi podatke o izmenjavah. Aplikacija naj nudi intuitiven uporabniški vmesnik za enostaven vpis vseh potrebnih podatkov o študijskih predmetih ter omogoča preverjanje in potrjevanje ustreznosti izbranih predmetov tako, da vsa komunikacija med študentom in koordinatorjem pri usklajevanju študijskega sporazuma poteka preko spletne aplikacije. Posebno pozornost pri razvoju aplikacije posvetite tudi varnosti, predvsem avtentikaciji uporabnikov in preprečevanju morebitnega nepooblaščenega dostopa do podatkov prijavljenih študentov. Razvita aplikacija naj bo zasnovana tako, da bo prilagodljiva zaslonom različnih ločljivosti in s tem uporabna tudi na mobilnih napravah.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jernej Bevk sem avtor diplomskega dela z naslovom:

Spletna aplikacija za upravljanje študijskih sporazumov študentov na izmenjavi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Alenke Kavčič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 24. januarja 2016 Podpis avtorja:

Zahvaljujem se mentorici, viš. pred. dr. Alenki Kavčič za vse koristne nasvete in napotke pri izdelavi diplomske naloge. Zahvala gre tudi moji družini za vso podporo v času študija.

Kazalo

Seznam uporabljenih kratic

- **AJAX** (angl. *Asynchronous JavaScript and XML*) - Skupina razvojnih tehnik za izdelavo dinamičnih spletnih strani.
- **CSS** (angl. *Cascading Style Sheets*) - Stilska predloga spletne strani.
- **HTML** (angl. *Hyper Text Markup Language*) - Označevalni jezik za izdelavo spletnih strani.
- **LDAP** (angl. *Lightweight Directory Access Protocol*) - Programski protokol za poizvedovanje imeniških storitev.
- **MySQL** (angl. *My Structured Query Language*) - Sistem za upravljanje z zbirkami podatkov.
- **PDF** (angl. *Portable Document Format*) - Odprt standard za izmenjavo elektronskih dokumentov.
- **PHP** (angl. *PHP Hypertext Preprocessor*) - Programski jezik za izdelavo spletnih strani.

Povzetek

Naslov: Spletna aplikacija za upravljanje študijskih sporazumov študentov na izmenjavi

V diplomskem delu je predstavljen razvoj spletne aplikacije, ki študentom v okviru programa Erasmus+ omogoča izpolnjevanje študijskih sporazumov. Na kratko je predstavljen trenutni potek izpolnjevanja ter zahteve aplikacije, ki so morale biti implementirane. Predstavljene so tudi tehnologije in knjižnice, ki so se med razvojem uporabljale. V poglavju o razvoju aplikacije je podrobneje razložen razvoj spletnega vmesnika in zaledne aplikacije, kjer so predstavljeni ključni elementi, ki jih je bilo potrebno razviti. Sledi poglavje o varnosti, kjer so opisani vsi varnostni pristopi, ki so nujni za ustrezno varnostno zaščito aplikacije in njenih podatkov. V poglavju Končne funkcionalnosti so predstavljene vse funkcionalnosti, ki so bile v okviru diplomskega dela implementirane. Diplomsko delo se zaključi s poglavjem o testiranju, kjer so podrobneje predstavljeni vsi koraki testiranja, ki so bili potrebni pred samim zaključkom razvoja.

Ključne besede: študijske izmenjave, PHP, SQL, izpolnjevanje obrazcev, odzivne spletne strani.

Abstract

Title: Web application for managing Learning Agreements of exchange students.

The thesis presents the development of web application that allows students within Erasmus+ program to prepare learning agreements. Current course of preparing learning agreements and demands, that needed to be implied, are briefly described. Thesis also presents technologies and libraries used during development. In the chapter of application development, the front-end and back-end development is explained in detail, where main elements that needed to be developed are described. Next chapter is about security, where all security approaches are presented, that are vital for proper security protection of the application and its data. Chapter final functionalities presents all functionalities implemented within this thesis. Document concludes with the chapter about testing, where all the testing steps, needed before the conclusion of development, are explained in detail.

Keywords: study exchanges, PHP, SQL, filling out forms, responsive design.

Poglavje 1

Uvod

Živimo v obdobju, ko si življenja brez spleta, pametnih telefonov in drugih elektronskih naprav pravzaprav ne moremo predstavljati. Nenehno se dogajajo veliki tehnološki premiki, ki nam skušajo ob natrpanem ritmu življenja posamezna opravila poenostaviti. To se opazi tudi pri izpolnjevanju različnih obrazcev. Vse več podjetij in ustanov tako skuša ročno izpolnjevanje papirnatih obrazcev nadomestiti s temu namenjenimi računalniškimi in spletnimi programi.

Ročno izpolnjevanje je izpostavljalo kar nekaj pomanjkljivosti. Glavna pomanjkljivost tovrstnih obrazcev je možnost nadaljne obdelave. Če so administratorji obrazcev želeli pridobljene podatke naprej uporabljati za izvajanje statistike ali druge obdelave, so morali podatke najprej prepisati v elektronsko obliko, kar jim je vzelo veliko časa. Prav tako so lahko problem predstavljali tudi napačni ali nečitljivi vnosi, kar je administratorjem po nepotrebnem oteževalo preverjanje podatkov.

Spletno izpolnjevanje obrazcev takšnih pomanjkljivosti nima. Izvajanje statistike nad izbranimi podatki je enostavno in hitro, saj so vsi podatki v elektronski obliki ter so zbrani na enem mestu (najpogosteje v podatkovni bazi). Takšen način izpolnjevanja obrazcev preprečuje tudi pojavitev napak pri vnosu, saj je večina spletnih aplikacij implementirana tako, da zazna napačen vnos in o tem pravočasno obvesti uporabnika. Izpolnjevanje in pre-

gledovanje poteka hitreje in učinkoviteje, kar je pravzaprav bistvo takšnega izpolnjevanja obrazcev. Tako kot vsaka spletna aplikacija ima tudi spletno izpolnjevanje obrazcev določene pomanjkljivosti. Ker so obrazci namenjeni širšemu krogu uporabnikov, od katerih zahtevamo veliko vnašanja, se pojavlja dvom o varnosti tovrstnih podatkov. Zato je potrebno področje varnosti dobro preučiti ter z uporabo dobrih varnostnih tehnik preprečiti vsak morebiten poskus napada na strukturo aplikacije.

V okviru diplomskega dela smo za potrebe Fakultete za računalništvo in informatiko Univerze v Ljubljani razvili spletno aplikacijo, ki študentom omogoča sestavljanje in popravljanje študijskih sporazumov za študijske izmenjave po programu Erasmus+. Aplikacija je poleg študentom namenjena tudi koordinatorjem študijskih izmenjav, ki lahko oddani študijski sporazum pregledajo, po potrebi komentirajo ter odobrijo ali zavrnejo. Razvita aplikacija tako omogoča hitrejšo in učinkovitejšo medsebojno komunikacijo, saj celotna komunikacija poteka preko spletne aplikacije.

V nadaljevanju diplomskega dela smo podrobneje predstavili celoten potek razvoja spletne aplikacije od zajema zahtev, načrtovanja in razvoja do testiranja. Posebno poglavje smo posvetili tudi varnosti, saj je to področje eno ključnih pri razvoju tovrstnih spletnih aplikacij.

Poglavje 2

Opis problema

2.1 Potek izpolnjevanja študijskih sporazumov

Dosedanji potek izpolnjevanja študijskih sporazumov sta sestavljala dva sklopa. V prvem sklopu je moral študent pripraviti študijski sporazum, kjer je določil, kakšne predmete bo na gostujoči instituciji opravljal. Prav tako je bilo potrebno določiti, katere domače predmete želi študent imeti priznane. Ker študentje za izpolnjevanje študijskih sporazumov niso uporabljali nobene spletne aplikacije, je bilo potrebno obrazce izpolnjevati ročno.

V drugem sklopu je študent izpolnil obrazec za morebitne spremembe, ki jih v študijskem sporazumu želi uveljavljati. Tukaj je lahko spremenil predmete na tuji ali domači instituciji ter spremenil predviden datum odhoda in prihoda. Tudi tukaj je bilo potrebno obrazec izpolnjevati ročno.

Takšen način izpolnjevanja pa ima kar nekaj slabosti. Poleg slabše preglednosti celotnega obrazca je potrebno izpostaviti tudi časovno potratnost. Ko študent ročno izpolni študijski sporazum, ga mora podpisanega dostaviti domači instituciji v pregled. Če je obrazec pravilno izpolnjen, potem večje časovne potratnosti ni. Težava pa lahko nastane pri napačno izpolnjenih obrazcih, saj mora v tem primeru študent študijski sporazum ponovno izpolnjevati ter prepisovati, pa še takrat lahko ne bo pravilno izpolnjen.

Pomembna prednost uporabe aplikacije je tudi možnost pregleda nad vsemi študijskimi sporazumi študentov in beleženje zgodovine dopolnitev vsakega sporazuma. Poleg tega omogoča, da je zadnja različica sporazuma vedno na voljo za pregled v aplikaciji.

2.2 Zahteve aplikacije

Prav z namenom poenostavitve obeh sklopov smo se odločili razviti spletno aplikacijo, ki bo študentom omogočala spletno izpolnjevanje ter popravljanje študijskih sporazumov. Aplikacija je seveda namenjena tudi administratorju, ki bo tako celotni študijski sporazum lahko pregledal v elektronski obliki.

Razviti je bilo potrebno spletno aplikacijo, ki bo študentu omogočila pripravo študijskega sporazuma v elektronski obliki. Pri tem študentu ni potrebno vnašati vseh podatkov o izmenjavi, saj administrator že znane podatke uvozi v aplikacijo preko datotek `.xlsx` in `.csv`. Aplikacija naj študentu omogoča tudi uveljavljanje sprememb, ki nastanejo med izmenjavo.

Administrator aplikacije lahko izpolnjene obrazce pregleda in jih po potrebi komentira, študijski sporazum in spremembe pa odobri ali zavrne. Administrator naj ima možnost pregleda že potrjenih študijskih sporazumov ter pregled nad statusi študijskih sporazumov vseh študentov, prijavljenih na izmenjavo.

Prav tako mora biti razvito samodejno generiranje dokumentov PDF v taki obliki, kot jo predpisuje program Erasmus+, ter njihovo tiskanje. Aplikacija mora biti odzivna, kar pomeni, da mora biti prikaz aplikacije prilagojen napravi, na kateri aplikacijo uporabljamo.

Poglavje 3

Uporabljene tehnologije in knjižnice

Med izdelavo spletne aplikacije smo se srečevali z različnimi tehnologijami. V sledečih podpoglavjih so na kratko predstavljene vse uporabljene tehnologije in knjižnice, s katerimi smo si pomagali pri realizaciji naše spletne aplikacije.

3.1 Tehnologije

3.1.1 HTML

Jezik HTML (angl. *Hyper Text Markup Language*) je označevalni jezik, ki predstavlja nekakšno ogrodje za izdelavo vseh spletnih strani [7]. Struktura spletne strani je predstavljena z uporabo ustreznih značk HTML. Stran se praviloma začne z značko `<html>`, ki predstavlja celoten dokument. Znotraj te značke je nato značka `<head>`, ki ponavadi vsebuje naslov spletne strani, meta podatke in vse potrebne knjižnice. Temu sledi značka `<body>`, znotraj katere je predstavljena dejanska vsebina, ki se bo uporabniku prikazala na spletni strani. Kot je razvidno iz izseka kode 3.1, lahko s pomočjo ustreznih značk HTML vsebino predstavljamo na različne načine. Z nekaj dodatnimi značkami lahko prikazujemo tabele, odstavke, sezname in podobne elemente.

```
<html>
  <head>
    <title>Sistem Erasmus+</title>
  </head>
  <body>
    <ul>
      <li>Domov</li>
      <li>Odjava</li>
    </ul>
  </body>
</html>
```

Izsek kode 3.1: Primer kode HTML z uporabo seznama.

3.1.2 CSS

CSS (angl. *Cascading Style Sheets*) je stilska predloga, ki nam določa, kakšne lastnosti ima posamezna značka HTML (izsek kode 3.2). V obstoječi dokument HTML jo lahko vključimo kot ločeno datoteko s končnico `.css`, lahko pa jo vključimo tudi neposredno v značko HTML [6]. Značkam lahko določamo tako statične lastnosti (npr. barva in velikost besedila, odmike in ozadje), kot tudi dinamične lastnosti (npr. barva besedila, ko se z miškinim kazalcem pomaknemo na izbrani element).

```
li:hover{
  color: red;
  text-decoration: underline;
}
```

Izsek kode 3.2: Primer dodajanja stilov.

3.1.3 JavaScript

JavaScript je objektni skriptni programski jezik, ki ga je podjetje Netscape razvilo z namenom, da programerju oziroma razvijalcu omogoči lažje razvijanje dinamičnih interaktivnih spletnih strani. S pomočjo jezika JavaScript lahko značkam HTML spreminjamo vrednosti, ne da bi bilo potrebno osveževati spletno stran, saj se koda izvaja na strani odjemalca. Kodo lahko v obstoječ dokument HTML vključimo kot ločeno datoteko (s končnico `.js`), lahko pa jo pišemo tudi neposredno v dokument HTML (znotraj značke `<script>`).

3.1.4 PHP

PHP (angl. *PHP Hypertext Preprocessor*) je strežniški programski jezik, ki se ga uporablja za razvoj dinamičnih spletnih strani. Za izvajanje programov v jeziku PHP potrebujemo interpreter PHP, ki teče na spletnem strežniku. Ta izvorno kodo, ki jo dobi kot vhod, pregleda, izvede ukaze v jeziku PHP in na izhod vrne generirano vsebino spletne strani [11]. PHP je nekakšna razširitev spletne strani HTML, saj ponuja več dinamičnosti [1, 2]. Uporaba jezika PHP nam omogoča pisanje lastnih funkcij (npr. branje podatkov iz obrazcev, odpiranje, branje in pisanje različnih datotek) kot tudi uporabo že obstoječih (npr. `date()`, `str_replace()` in `usort()`). Koda se praviloma nahaja znotraj oznak `<?php` in `?>` (izsek kode 3.3).

```
1 <?php
2   $variable = "Pozdravljen, Svet!";
3   for ($i = 1; $i <= 5; $i++){
4     echo $i." - ".$variable;
5     echo "<br>";
6 ?>
```

Izsek kode 3.3: Primer izpisa na zaslon z uporabo jezika PHP.

3.1.5 MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami, ki za delo s podatki uporablja jezik SQL (angl. *Structured Query Language*). Deluje po principu odjemalec - strežnik, kar pomeni, da je podatkovna baza nameščena na strežniku, do nje pa lahko dostopa en ali več odjemalcev [12]. Do podatkov v podatkovni bazi dostopamo z uporabo ustreznih poizvedbenih ukazov (izsek kode 3.4).

```
1 SELECT *
2 FROM erasmus_admin
3 WHERE tehnicna_pomoc = 1
4 ORDER BY priimek
```

Izsek kode 3.4: Primer poizvedbe SQL.

3.1.6 AJAX

AJAX (angl. *asynchronus JavaScript and XML*) je skupina več spletnih razvojnih tehnik, ki so namenjene izdelavi interaktivnih spletnih strani. Uporaba tehnologije AJAX omogoča asinhrono izmenjavo podatkov med odjemalcem in strežnikom brez potrebe po vnovičnem nalaganju vsebine spletne strani (izsek kode 3.5).

```
1 $.ajax({
2    url:"ajax.php",
3    type:"GET",
4    success:function(data){
5        alert(data);
6     }
7 });
```

Izsek kode 3.5: Primer uporabe tehnologije AJAX.

3.2 Knjižnice

3.2.1 jQuery

JQuery velja za eno izmed najbolj razširjenih in priljubljenih knjižnic, napisanih v skriptnem jeziku JavaScript. Uporaba knjižnice omogoča enostavnejše programiranje (izsek kode 3.6) in dostopanje do elementov strani. Prav tako lahko z uporabo knjižnjice jQuery ustvarimo animacije, poizvedbe AJAX ter sprožilne funkcije. JQuery deluje pod licenco MIT (*Massachusetts Institute of Technology*), kar pomeni, da je koda knjižnice prosto dostopna in jo lahko uporabljamo tako zasebno kot tudi v komercialne namene.

```
1 $(document).ready( function() {
2   $('.phone-menu').click( function () {
3       $('.mobile-menu').toggle();
4   });
5 });
```

Izsek kode 3.6: Primer uporabe knjižnice jQuery.

3.2.2 Simple XLSX

Simple XLSX je knjižnica, napisana v strežniškem jeziku PHP. Namenjena je razčlenjevanju in branju podatkov iz datotek, napisanih v programu Microsoft Excel. Knjižnica omogoča branje vrednosti posameznih vrstic in stolpcev iz razpredelnice. Knjižnica Simple XLSX deluje pod licenco MIT (*Massachusetts Institute of Technology*).

3.2.3 XLSX Writer

XLSX Writer je knjižnica, napisana v strežniškem jeziku PHP. Namenjena je ustvarjanju novih dokumentov, ki jih podpira program Microsoft Excel.

Knjižnica XLSX Writer deluje pod licenco MIT (*Massachusetts Institute of Technology*).

3.2.4 mPDF

MPDF je knjižnica, napisana v strežniškem jeziku PHP. Uporablja se za ustvarjanje novih dokumentov PDF (angl. *Portable Document Format*). S pomočjo knjižnice mPDF lahko v dokument vstavljamo slike, tabele in ostale elemente, s podporo za stilske predloge pa lahko tem elementom spremenimo tudi vizualni prikaz. Knjižnica deluje pod licenco GNU (*General Public License*), ki nam dovoljuje uporabo te knjižnice v zasebne in komercialne namene.

3.2.5 PHPMailer

PHPMailer je knjižnica, napisana v strežniškem jeziku PHP. Zaradi njene preprostosti in hitrosti velja za eno izmed najbolj priljubljenih knjižnic PHP s področja pošiljanja elektronskih sporočil. Z njeno pomočjo lahko elektronskemu sporočilu poljubno spreminjamo stil in postavitev. Knjižnica deluje pod licenco GNU (*General Public License*).

3.2.6 Bootstrap

Ko govorimo o izdelavi odzivnih spletnih strani, skoraj ne moremo mimo knjižnice Bootstrap. Knjižnico Bootstrap so razvili razvijalci socialnega omrežja Twitter, deluje pa pod licenco MIT (*Massachusetts Institute of Technology*). Bootstrap je pravzaprav ogrodje, ki temelji na jezikih HTML, CSS in JavaScript [3]. Glavna komponenta načrtovanja s pomočjo knjižnice Bootstrap je tabela z dvanajstimi stolpci. Z uporabo te tabele lahko določamo, kako se bo vsebina spletne strani prikazovala na posameznih napravah. Kot je razvidno na primeru iz izseka kode 3.7, smo tabelo z dvanajstimi stolpci združili v tri dele. Prvi in tretji del tabele zasedata vsak po eno četrtino širine

celotnega razpoložljivega prostora spletne strani, drugi del pa zaseda preostanek. Tako sestavljena programska koda se nam bo na namiznem računalniku prikazala kot tabela s tremi stolpci, na mobilnem telefonu pa bo prikaz zaradi manjšega zaslona razporejen drugače. Tam se bodo posamezni stolpci ustrezno pomanjšali in prikazali drug pod drugim.

```
1 <div class="container">
2   <div class="row">
3       <div class="col-sm-3"> Stolpec 1 </div>
4       <div class="col-sm-6"> Stolpec 2 </div>
5       <div class="col-sm-3"> Stolpec 3 </div>
6   </div>
7 </div>
```

Izsek kode 3.7: Primer uporabe knjižnice Bootstrap.

Bootstrap pravzaprav ne deluje po principu zaznavanja naprave. Knjižnica stalno preverja dimenzijo brskalnika, v katerem spletno stran gledamo, in temu primerno popravlja izgled. Tako je prav mogoče, da se bo postavitev elementov spletne strani razlikovala tudi na različnih osebnih računalnikih, saj je celoten prikaz pogojen z velikostjo zaslona.

```
1 <button type="button" class="btn btn-success btn-lg">
2   Velik gumb
3 </button>
4 <button type="button" class="btn btn-danger btn-xs">
5   Zelo majhen gumb
6 </button>
```

Izsek kode 3.8: Primer uporabe vnaprej določenih razredov knjižnice Bootstrap.

Uporaba knjižnice Bootstrap je priljubljena tudi zaradi nekaterih vnaprej določenih razredov, ki omogočajo lažje in hitrejše oblikovanje elementov spletnih strani. Kot je razvidno iz izseka kode 3.8, nam za izdelavo gumbov s pomočjo knjižnice Bootstrap ni potrebno pisati dodatnih stilskih predlog, ki bi določale barvo ali velikost gumbov, saj to lahko uredimo z uporabo vnaprej določenih razredov. Tako sestavljena koda nam generira dva gumba. Prvi je večji (razred `.btn-lg`) in zelene barve (razred `.btn-success`), drugi pa majhen (razred `.btn-xs`) in rdeče barve (razred `.btn-danger`).

Poglavje 4

Razvoj aplikacije

Preden smo lahko začeli z dejanskim razvojem spletne aplikacije, je bilo potrebno določiti arhitekturo sistema ter opredeliti način poteka razvoja aplikacije.

Ker gre pri naši aplikaciji za klasično spletno aplikacijo, smo se odločili za uporabo arhitekture odjemalec - strežnik. Na strežniški strani teče zaledni del aplikacije, na odjemalčevi strani pa se v spletnem brskalniku izvaja interaktivni spletni vmesnik aplikacije.

Tudi glede načina razvoja aplikacije ni bilo večjih pomislekov. Ker je bila izdelava spletne aplikacije dodeljena zgolj eni osebi, smo se odločili za delno prilagojen inkrementalni pristop k razvoju. Tako smo že med samim razvojem tudi testirali delovanje na posameznih napravah in brskalnikih, s čimer smo si olajšali delo pri zaključnih testiranjih.

Celoten del razvoja spletne aplikacije smo s tehničnega vidika razdelili na dva dela. Prvi del je razvoj spletnega vmesnika, ki predstavlja vizualno stran, ki je vidna uporabniku in s katero tudi komunicira. Tukaj se določi postavitev glavnih elementov, barve in velikosti pisav. Tukaj se v bistvu določi vse, kar lahko uporabnik spletne aplikacije vidi. Drugi del je razvoj zalednega dela aplikacije. Sem spada vse tisto, česar uporabnik ne vidi, a je vseeno ključnega pomena pri delovanju aplikacije. Ta del skrbi za vzpostavljanje povezav s podatkovnimi bazami in črpanjem podatkov iz njih, postavljanje

sej, pošiljanje sporočil in drugih funkcionalnosti spletne aplikacije [8].

4.1 Razvoj spletnega vmesnika

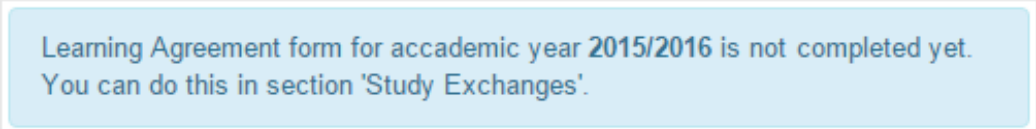
Kot smo že zgoraj omenili, je ta del namenjen določitvi vizualne podobe spletne aplikacije. Tukaj se določi vse, kar je uporabniku vidno. Ker je izbrana spletna aplikacija resnega značaja, je bilo potrebno temu primerno prilagoditi njen videz. Grafično predlogo smo poskušali čimbolj približati že obstoječi grafični predlogi, ki jo fakulteta uporablja v študijskem informacijskem sistemu. Pri izdelovanju in grafičnem oblikovanju vizualnega dela smo uporabljali jezika HTML in CSS.

Da bi spletna aplikacija ustrezala vsem načelom dobrega načrtovanja, smo si pri razvoju pomagali z desetimi Nielsenovimi principi [4].

4.1.1 Obveščanje uporabnika o stanju sistema

Ta Nielsenov princip govori o tem, da je potrebno uporabnika sproti obveščati o trenutnem stanju sistema. V naši spletni aplikaciji smo ta princip uporabili pri implementaciji povratnih informacij aplikacije.

Kot je razvidno s slike 4.1, smo v spletni aplikaciji implementirali obvestila o trenutnem statusu študijskega sporazuma, ki na podlagi podatkov iz podatkovne baze prikažejo ustrezno obvestilo. Obvestila se prikazujejo tako na strani študenta, kot tudi na strani administratorja.



Slika 4.1: Primer obvestila o trenutnem statusu študijskega sporazuma.

Poleg obvestil o stanju študijskih izmenjav smo implementirali tudi obveščanje preko elektronskih sporočil o ustreznosti izpolnjenih obrazcev. Obvestila smo

implementirali tudi pri obveščanju uporabnika ob morebitnih napačnih vnosih v polja obrazca.

4.1.2 Prilagoditev realnemu svetu

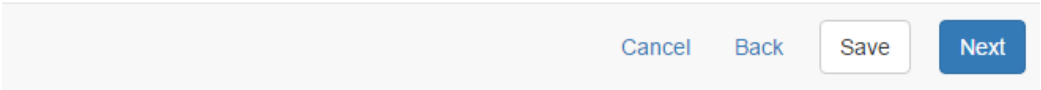
Ta princip govori o tem, da naj bo vsebina spletne aplikacije prikazana v uporabniku razumljivem jeziku in z uporabo izrazov, ki jih uporabnik aplikacije že pozna. Prav tako morajo biti posamezni sklopi spletnega obrazca zasnovani enako kot v papirnem obrazcu, ki ga študentje že poznajo.

Uporabniški vmesnik spletne aplikacije je v celoti v angleškem jeziku. Tako uporabnika opozorimo, da naj v vnosna polja vpisuje angleške izraze, saj mora biti končni obrazec študijskega sporazuma sestavljen v angleščini. Poleg tega je aplikacija namenjena tudi tujim študentom fakultete, ki slovenščine ne poznajo dovolj dobro, da bi jo lahko suvereno uporabljali pri izpolnjevanju obrazcev.

4.1.3 Uporabnikov nadzor

Uporabnik lahko na določeno podstran spletne aplikacije pride tudi pomotoma. Tretji Nielsenov princip govori o tem, da moramo v takem primeru uporabniku omogočiti svobodno vračanje nazaj ali preklicati trenutno akcijo.

V spletni aplikaciji smo pod izpolnjevalnimi obrazci dodali navigacijsko pasico (slika 4.2), ki omogoča uporabniku prosto sprehajanje naprej in nazaj po straneh. Prav tako ima uporabnik možnost preklica trenutnega dela, s čimer se pobrišejo vsi neshranjeni obrazci, uporabnika pa sistem preusmeri na začetno stran.



Slika 4.2: Primer navigacijske pasice na strani z obrazci.

Uporabnik lahko prav tako uporablja privzete navigacijske gumbe brskal-

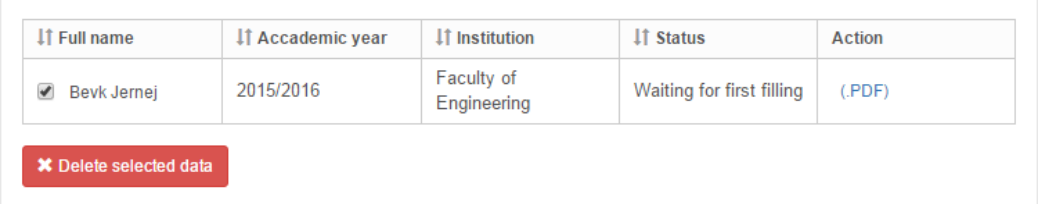
nika, vendar je zaradi vidika varnosti pri podstraneh, namenjenih izpolnjevanju obrazcev, potrebno ponovno pošiljanje podatkov.

4.1.4 Konsistentnost in standardi

Princip konsistentnosti in standardov pojasnjuje, da morajo imeti deli, ki pomenijo podobno, enake lastnosti.

V spletni aplikaciji smo tako za gumbe, ki imajo enake funkcionalnosti, določili enako barvo, enako velikost in enako obliko. Z rdečo barvo smo dodatno poudarili gumbe in elemente strani, ki od uporabnika zahtevajo posebno pozornost (slika 4.3).

Prav tako smo se držali načel, da mora biti vsebina na posameznih podstraneh podobno razdeljena ter da morajo biti naslovi povsod enake velikosti, barve in pisave.



Slika 4.3: Primer gumba, ki od uporabnika zahteva dodatno pozornost.

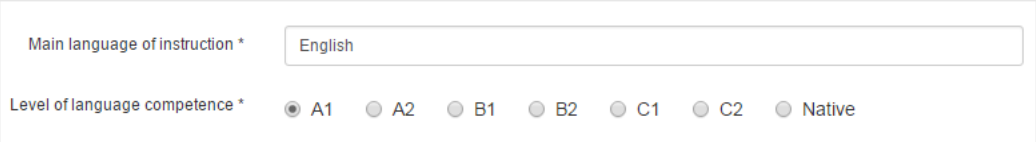
4.1.5 Izogibanje napakam

Peti Nielsenov princip dobrega načrtovanja pojasnjuje, da je eden izmed najbolj ključnih korakov tudi preprečevanje napačnih vnosov in akcij uporabnika. Pri načrtovanju spletne aplikacije smo to tudi upoštevali. Uporabili smo kratka in jasna imena polj, ki uporabniku takoj povejo, kaj je potrebno v polje vpisati.

Ker imamo v obrazcih tudi taka polja, ki imajo omejen nabor vrednosti, smo v ta namen implementirali izbirne gumbe (angl. *radio button*) (slika 4.4).

Pri poljih, namenjenih izbiri ustreznega semestra predmeta, smo raje implementirali izbirne sezname (angl. *drop-down list*), in tako poskrbeli za skladen izgled strani.

S temi pristopi smo želeli od uporabnika spletne aplikacije zahtevati čim manj vnašanja in s tem zmanjšati možnosti za napačne vnose.



Slika 4.4: Primer implementacije izbirnih gumbov.

4.1.6 Bolje prepoznati kot zapomniti

Vedno moramo skrbeti, da si uporabniku med obiskom spletne strani ni potrebno zapomniti vseh podatkov. Pomembno je, da so ključni elementi strani na vidnem in vedno na istem mestu (v naši spletni aplikaciji je to npr. navigacijski menu zgoraj desno).

Ker moramo obremenitev spomina uporabnika minimizirati in ker so nekateri njegovi podatki že znani, smo omogočili uvoz teh podatkov za vse uporabnike. Tako lahko administrator spletne aplikacije ob dodeljevanju dostopa študentom uvozi ustrezno datoteko s podatki o študentih (uvoz podpira formata `.xlsx` in `.csv`). Nato sistem prebere tiste podatke, ki jih potrebuje, in jih zapiše v podatkovno bazo. Tako od uporabnika zahtevamo le tiste vnose, ki še niso znani.

Ker je celotni obrazec zelo obsežen in se ta lahko še podaljšuje (npr. če želimo dodati dodatne predmete ali zamenjati že izbrane in potrjene predmete), smo izpolnjevanje razdelili na več podstrani. S tem smo vsaki strani dodali preglednost in hitrejše prehajanje po posameznih razdelkih. Pred zaključkom izpolnjevanja se uporabniku prikaže tudi predogled obrazca, kjer lahko uporabnik preveri vsa izpolnjena polja. Če je prišlo do napačnega vnosa oz. če želi uporabnik popraviti vnos, lahko to popravi tako, da se pre-

prosto z uporabo navigacijske pasice vrne na prejšnjo stran in popravi želene vnose. Ostali podatki pri tem ostanejo nespremenjeni.

4.1.7 Fleksibilnost in učinkovitost

Princip fleksibilnosti in učinkovitosti govori o tem, da mora biti stran zgrajena tako, da bo prilagodljiva tako izkušenemu uporabniku kot popolnemu začetniku. Prav tako govori o tem, da naj ima izkušen uporabnik možnost uporabe bližnjic za hitrejše delo. V naši spletni aplikaciji smo omogočili navigacijo med posameznimi vnosnimi polji in stranmi tako s pomočjo miške (klik v vnosno polje ali na gumb) kot z uporabo tipkovnice (bližnjice za hitrejše delo).

Poskrbeli smo tudi, da je izgled fleksibilen, kar pomeni, da je aplikacija pravilno prikazana ne glede na vrsto naprave, ki jo uporabljamo. V ta namen smo si pomagali z ogrodjem Bootstrap, s katerim smo lahko hitro in enostavno nadzorovali način prikaza na posamezni napravi.

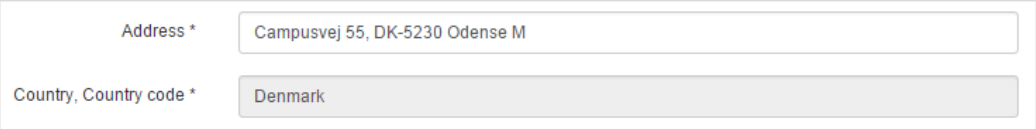
Prav tako mora biti spletna stran prilagojena posameznemu tipu uporabnika. Za potrebe naše aplikacije smo tako načrtovali dve različni vrsti uporabnikov (administrator in študent), ki imata različno osnovno stran. Tako prijavljen študent nima na voljo enakih funkcionalnosti, kot jih ima administrator.

4.1.8 Estetika in minimalizem

Princip estetike in minimalizma določa, da je dobra spletna stran tista stran, ki je preprosta, elegantna, minimalistična in uporabniku nudi le tiste informacije, ki jih res potrebuje.

Splošno znano je, da so naši možgani navajeni na simetrijo in da vsakršen nered v naših možganih povzroča stres [10]. Zato smo se tudi v naši spletni aplikaciji držali načela estetike in minimalizma. Tako smo za izgled strani uporabljali študentom že poznano postavitev iz študijskega informacijskega sistema. Uporabili smo nežne, ne preveč kontrastne barve, saj nismo želeli

po nepotrebnem pozornost preusmerjati na dele, ki je ne potrebujejo. Gumbi so svetlih barv z ikonami, ki uporabniku pojasnjujejo, čemu je gumb namenjen. Prav tako so svetlih barv tudi vnosna polja, ki so vsa enako velika in ustrezno grupirana. Polja, ki jih ni mogoče urejati, so tudi jasno poudarjena s temnejšim odtenkom in onemogočena za urejanje (slika 4.5).



Slika 4.5: Primer omogočenih in onemogočenih polj.

Stran smo želeli ohraniti čim bolj preprosto, zato smo uporabili minimalno število podstrani. Nekatere dele strani smo načrtovali celo tako, da se prikažejo zgolj ob kliku na gumb. Tako na primer izbira gumba za uvoz študentov prikaže nov razdelek z dodatnimi možnostmi.

Prikazali smo samo tiste informacije, ki so za študenta in administratorja pomembne, saj potrebe po dodajanju nepomembnih informacij tudi ni.

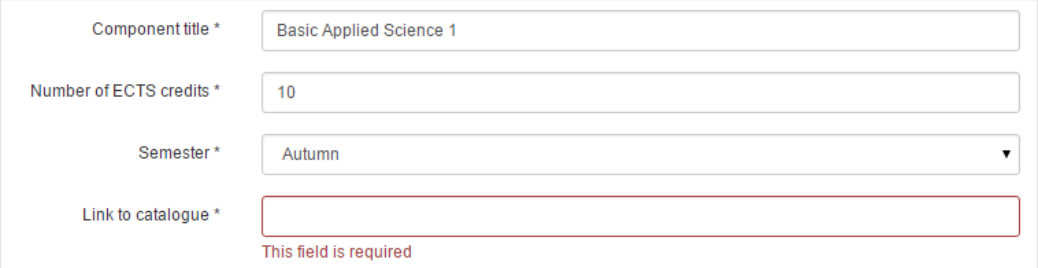
4.1.9 Obveščanje o napakah in njihovo reševanje

Uporabnika spletne strani je potrebno sproti obveščati o morebitnih napačnih vnosih. Princip obveščanja o napakah in njihovega reševanja govori o tem, da morajo biti sporočila o napakah jasna, uporabniku razumljiva in točna. Prav tako mora uporabnik vedeti, kaj mora popraviti in kako. V naši spletni aplikaciji smo definirali tri vrste opozoril.

Prva vrsta opozoril o napakah so klasična opozorila, ki se prikazujejo ob napačnem vnosu v obrazcu (npr. ko študent ne izpolni obveznega polja ali ko v polje vnese neveljavno vrednost). Ko se obvestilo o napaki pojavi, se nadaljevanje izpolnjevanja obrazca onemogoči, dokler uporabnik polja ne popravi (slika 4.6).

Druga vrsta opozoril so prilagojena opozorila. To so opozorila, ki jih administrator določi ob pregledovanju posameznega obrazca. S tehničnega

vidika so lahko vsa polja ustrezno izpolnjena, a so vsebinsko neprimerna. Zato ima administrator možnost označevanja, kateri sklop obrazca je napačen in kako naj ga študent popravi.



Slika 4.6: Primer opozorila ob napačnem vnosu.

Tretja vrsta opozoril so povratna opozorila. To so tista opozorila, ki se študentu prikažejo po tem, ko je obrazec že oddan v pregled. Ta opozorila se prikažejo, npr. ko je administrator zavrnil študentov obrazec. Tako študent dobi obvestilo, da je obrazec potrebno ponovno izpolniti. Pri ponovnem izpolnjevanju obrazca se tako študentu sproti prikazujejo obvestila, kje je potreben popravek.

4.1.10 Pomoč in dokumentacija

Zadnji Nielsenov princip priporoča dostop do pomoči in dokumentacije. Naša spletna aplikacija posebnega razdelka, namenjenega pomoči in dokumentaciji, nima, saj je spletna aplikacija enostavna in intuitivna za uporabo, vsebina pa je predstavljena jasno in razločno, zato posebna dokumentacija niti ni potrebna.

Ker pa se lahko vseeno pojavi kakšna težava ali nejasnost s strani uporabnikov, smo v ta namen implementirali možnost pošiljanja elektronskega sporočila tehnični podpori (slika 4.7).

Slika 4.7: Povezava do tehnične podpore.

4.2 Razvoj zaledne aplikacije

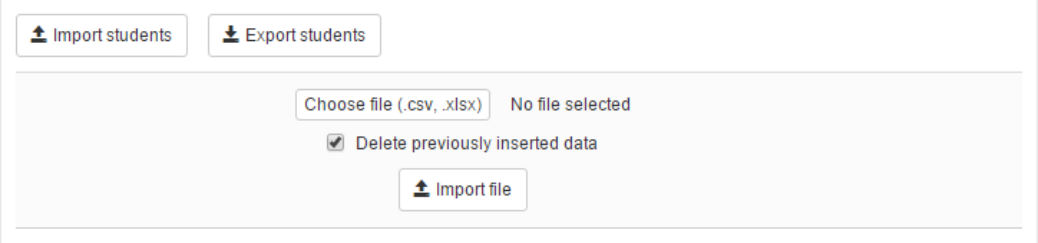
Ko je bil vizualni del spletne aplikacije zgrajen, smo lahko pričeli z dodajanjem različnih funkcionalnosti. Potrebno je bilo zasnovati podatkovno bazo, ki jo potrebujemo za shranjevanje in posodabljanje podatkov. Prav tako smo morali implementirati uvoz in izvoz podatkov iz različnih vrst datotek ter zgraditi programsko kodo, ki shranjene podatke v pravilnem formatu preslika v dokument PDF. V nadaljevanju so predstavljene nekatere glavne funkcionalnosti sistema, ki jih je bilo potrebno razviti.

4.2.1 Uvoz / izvoz podatkov

Če želi administrator študentom omogočiti prijavo v sistem, mora podatke o študentih najprej uvoziti. V ta namen smo implementirali možnost uvoza in izvoza podatkov. Spletna aplikacija podpira uvoze datotek formata `.xlsx` in `.csv`, ki morajo biti ustrezno razčlenjene. Ko je uvoz datoteke uspešno izveden, se podatki prenesejo v podatkovno bazo, s čimer določimo, kateri študenti imajo dostop do sistema. Kot lahko vidimo na sliki 4.8, se administratorju ob kliku na gumb za uvoz podatkov prikaže obrazec, ki vsebuje polje za izbor datoteke. Obrazec prav tako ponuja možnost izbora, ali naj se dosedanji podatki v podatkovni bazi ohranijo ali ne. Če izberemo izbris podatkov, potem jih sistem najprej izbriše in nato uvozi nove. Če pa želimo obstoječe podatke obdržati, potem sistem pri uvozu to upošteva in vnosov, ki so že v podatkovni bazi, ne uvozi.

Administrator ima tudi možnost izvoza podatkov, s čimer podatke iz

podatkovne baze izvozimo v datoteke ustreznih formatov in tako omogočimo nadaljno manipulacijo nad podatki.



Slika 4.8: Obrazec za uvoz podatkov o študentih.

4.2.2 Generiranje datotek PDF

Ko študent v naši spletni aplikaciji uspešno izpolni vse obrazce in ko jih administrator tudi potrdi, je potrebno iz vnešenih podatkov ustvariti datoteko PDF. Ustvarjeno datoteko lahko nato študent natisne in podpisano dostavi koordinatorju v podpis. Prav zato je generiranje datotek PDF ključnega pomena pri delovanju aplikacije.

V spletni aplikaciji obstajata dve različni vrsti obrazcev. Prvi obrazec je tisti obrazec, ki ga študent izpolni pred začetkom izmenjave. V njem študent vnese osebne podatke, podatke o domači in gostujoči instituciji, kontaktne podatke odgovornih oseb ter obdobje, v katerem naj bi izmenjava potekala. Študent tukaj tudi sestavi predmetnik, ki ga bo opravljal na gostujoči instituciji, ter določi, katere predmete na domači instituciji želi imeti priznane. Ko študent uspešno izpolni obrazec, se prične proces preslikave podatkov v datoteko PDF. Proces dokumentu vstavi ustrezno glavo, nato pa z uporabo tabel izpiše podatke iz obrazca. Proces se zaključi z dodanim razdelkom za podpise vseh sodelujočih (študent in odgovorni osebi na domači in gostujoči instituciji) in oštevilčenjem posameznih strani.

Ker pa mora biti končni dokument tudi stilsko dovršen, je potrebno pred generiranjem uvoziti potrebne stilske predloge, kjer določimo, kako naj se

posamezen element dokumenta prikaže. Ko je dokument uspešno sestavljen, ga pravilno poimenujemo ter shranimo (izsek kode 4.1).

Drugi obrazec je tisti obrazec, ki ga študent izpolni v primeru, ko želi uveljavljati določene spremembe v že izpolnjenemu prvem obrazcu. Tukaj študent označi, kaj bi rad spremenil. Lahko zamenja predmete na domači in tuji instituciji ter spremeni obdobje, v katerem naj bi izmenjava potekala. Prav tako lahko spremeni tudi odgovorno osebo na gostujoči instituciji.

Po uspešno izpolnjenem obrazcu je potrebno vnovično generiranje datoteke PDF. V tem primeru v že obstoječo datoteko (datoteko, ki smo jo ustvarili ob oddaji prvega obrazca) vstavimo vse izbrane spremembe ter znova dodamo razdelek za podpise vseh sodelujočih. Ko je dokument posodobljen, ga shranimo.

```
1 <?php
2 $mpdf = new mPDF();
3 $vsebina = "...";
4 $stil = file_get_contents('style.css');
5 $mpdf -> WriteHTML($stil , 1);
6 $mpdf -> WriteHTML($vsebina , 2);
7 $mpdf -> Output('../pot/datoteka.pdf' , 'F');
8 ?>
```

Izsek kode 4.1: Primer generiranja dokumenta PDF.

Teoretično lahko študent obrazec za uveljavljanje sprememb izpolni poljubnokrat (vendar je pred vsako spremembo potrebno potrjevanje s strani administratorja), zato se lahko generiranje dokumenta PDF izvede večkrat. Končna oblika dokumenta je tako sestavljena iz obrazca, ki ga je študent izpolnil pred študijsko izmenjavo, temu pa sledi ustrezno število obrazcev, ki jih je študent izpolnil, ko je želel uveljavljati spremembe.

4.2.3 Medsebojna komunikacija

Ena izmed ključnih lastnosti aplikacije, ki jih je bilo potrebno implementirati, je medsebojna komunikacija med študenti in administratorjem. Potrebno je bilo razviti programsko kodo, ki bo administratorju sporočila, kdaj je kakšen obrazec oddan. Prav tako mora študent biti obveščen, ali je administrator odobril ali zavrnil izpolnjen obrazec. V ta namen je bilo potrebno najprej definirati vsa možna stanja, v katerih se lahko znajde študent, in jim določiti unikatne številke:

- Stanje, ko noben obrazec ni izpolnjen (številka -2).
- Stanje, ko so obrazci poslani v pregled (številka 1).
- Stanje, ko je obrazce potrebno popraviti (številka -1).
- Stanje, ko so obrazci pravilno izpolnjeni (številka 2).

Tako se vsakič, ko študent izpolni obrazec, v podatkovno bazo pripiše unikatna številka (številka = 1). Podobno velja tudi pri administratorju. Če obrazec zavrne, se pripiše številka -1, v nasprotnem primeru številka 2.

Sistem ob vsaki prijavi pokliče ustrezne podporne funkcije, ki glede na vrsto uporabnika preverjajo posamezne atribute. Če se v aplikacijo prijavi administrator, potem sistem preveri, ali kakšen obrazec čaka na pregled (številka = 1) in to ustrezno sporoči. Če pa se v aplikacijo prijavi študent, potem sistem preveri, ali je kakšen obrazec napačno izpolnjen (številka = -1) ali pa je bil potrjen (številka = 2). Tako ima uporabnik vedno vpogled v trenuten potek izpolnjevanja.

Poleg prikazovanja sporočil preko sistema smo implementirali tudi samodejno pošiljanje elektronskih sporočil, ki se pošiljajo ob zaključku posameznih akcij. Ko študent izpolni obrazec, se pošljeta dve sporočili. Prvo sporočilo je poslano študentu in ga obvesti, da je bil obrazec uspešno oddan v pregled. Drugo sporočilo je poslano administratorju (če je administratorjev več, se sporočilo pošlje vsem) in ga obvesti, da je študent oddal obrazec v pregled. Sporočila se pošiljajo tudi ob administratorjevem pregledovanju. Ko

administrator zavrne obrazec, študent prejme elektronsko sporočilo, kjer je obveščen, da je potrebno obrazec popraviti. Če pa administrator obrazec odobri, se študentu pošlje sporočilo s potrdilom in nadaljnjimi navodili.

4.2.4 Preverjanje uporabnikov

Ko administrator v sistemu uvozi seznam študentov, se le ti lahko prijavijo v aplikacijo. Zato je bilo potrebno razviti programsko kodo, ki skrbi za varno preverjanje študentovih prijavnih podatkov. Zaradi preprostosti celotnega poteka prijave smo uporabili sistem enotne prijave, ki ga študenti že uporabljajo za dostop do drugih spletnih strani fakultete (študijski informacijski sistem in spletna učilnica). Pri realizaciji te funkcionalnosti smo si pomagali s protokolom LDAP. Protokol LDAP (angl. *Lightweight Directory Access Protocol*) je programski protokol, ki nam dovoljuje iskanje in spreminjanje imeniških storitev [9]. Ko se študent skuša prijaviti v aplikacijo, sistem najprej preveri, ali ima študent sploh pravico do prijave (ali se njegovo uporabniško ime nahaja med uvoženimi podatki), šele nato se sproži postopek avtentikacije preko protokola LDAP.

```
1  <?php
2
3  $ldapconn = ldap_connect($adServer, $ldapport)
4    or die("LDAP napaka.");
5
6  $ldapbind = ldap_bind($ldapconn, $username, $password);
7
8  if ($ldapbind) {
9    $result = ldap_search($ldapconn,$ldaptree,$filter,$attr)
10     or die ("Search failed: ".ldap_error($ldapconn));
11
12   $data = ldap_get_entries($ldapconn, $result);
13 }
```

```
14
15 ldap_close($ldapconn);
16 ?>
```

Izsek kode 4.2: Poenostavljen primer uporabe protokola LDAP.

Za lažje razumevanje delovanja protokola LDAP si poglejmo izsek programske kode 4.2. Prvi korak, ki ga moramo izvesti, je vzpostavitev povezave z ustreznim imeniškim strežnikom. To storimo z ukazom `ldap_connect()`, kateremu kot parametra dodamo naslov, kjer se imeniški strežnik nahaja, ter ustrezna vrata. Ko smo s strežnikom uspešno povezani, lahko z ukazom `ldap_bind()` preverimo, ali v strežniku obstaja vnos s predvidenim uporabniškim imenom in geslom. Če vnosa ni, vrne ukaz stanje `false`. Tedaj vemo, da sta uporabniško ime in geslo napačna, zato lahko o tem obvestimo uporabnika. Če pa je ukaz našel ustrezen vnos, potem moramo z ukazom `ldap_search()` zadetke filtrirati in tako preveriti, ali se uporabnik nahaja v ustreznem imeniškem poddirektoriju (v naši spletni aplikaciji so to vsi študenje Fakultete za računalništvo in informatiko). Ko so filtri uspešno postavljeni, z ukazom `ldap_get_entries()` preberemo podatke, ki se nahajajo v vnosu, in jih poljubno obdelamo. Če so vsi koraki uspešno izvedeni, potem ima študent omogočen dostop do aplikacije in je tako v aplikacijo tudi prijavljen.

Seveda pa morajo poleg študentov imeti tudi administratorji omogočeno prijavo v sistem. Ker je administratorjev lahko poljubno in ni nikjer predpisano, kakšna uporabniška imena in gesla morajo uporabljati, smo se za shranjevanje podatkov o administratorjih odločili za podatkovno bazo MySQL. V njej hranimo različne administratorjeve podatke, kot so uporabniško ime, geslo, ime, priimek ter zastavice, ki povejo, kakšna sporočila lahko prejema. Ne glede na to, da za prenos podatkov med odjemalcem in strežnikom uporabljamo varno povezavo SSL (angl. *Secure Sockets Layer*), je bilo potrebno za varnost administratorjevega gesla dodatno poskrbeti. Več o varnosti podatkov v aplikaciji je opisano v naslednjem poglavju.

Poglavje 5

Varnost

Glavna funkcionalnost naše aplikacije je elektronsko izpolnjevanje obrazcev, katerih vsebina se shranjuje v podatkovno bazo. Aplikacija tako od uporabnika zahteva večje število vnosov, s čimer je aplikacija toliko bolj izpostavljena morebitnim poskusom napada. Varnost je pomembna tudi zato, ker v bazi shranjujemo osebne podatke študentov.

Za varnost je bilo tako potrebno dodatno poskrbeti, s čimer smo želeli zagotoviti, da ima aplikacija ustrezni obrambni mehanizem, s katerim preprečimo nepooblaščenim osebam dostop do podatkov in sistema [5].

V nadaljevanju sta predstavljeni dve najpogostejši obliki napada na spletne strani ter varnostne tehnike, ki smo jih za preprečevanje napadov uporabili pri razvoju. Predstavljene so tudi ostale varnostne tehnike, s katerimi smo še dodatno poskrbeli za varnost aplikacije in njenih podatkov.

5.1 Vrivanje stavkov SQL

Vrivanje stavkov SQL (angl. *SQL injection*) velja za napadalcem enega izmed najbolj priljubljenih napadov na spletne strani. Napad poteka tako, da napadalec v vnosno polje vnese zlonamerno kodo, s katero skuša prevzeti nadzor nad delovanjem podatkovne baze. Če je za varnost na spletni strani slabo poskrbljeno, lahko napadalec v trenutku pobriše celotno podatkovno

bazo, s tem pa oškodovanec izgubi vse zaupne podatke.

Za lažje razumevanje poteka napada smo enostaven primer prikazali v izseku kode 5.1.

```
1 //primer stavka SQL pred napadom
2 SELECT * FROM USERS WHERE name = ''
3
4 //primer poskusa napada
5 a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't
6
7 //primer stavka SQL po napadu
8 SELECT * FROM USERS WHERE name = 'a';DROP TABLE users; SELECT
    * FROM userinfo WHERE 't' = 't'
```

Izsek kode 5.1: Primer napada z vrivanjem stavkov SQL.

Predpostavimo, da obstaja neka spletna stran, ki od uporabnika zahteva, da v vnosno polje vnese svoje ime. Ko uporabnik potrdi vnos, se izvede stavek SQL, ki v podatkovni bazi poišče uporabnika s takšnim imenom (druga vrstica v izseku kode 5.1). Če za varnost na spletni strani ni dovolj dobro poskrbljeno, lahko napadalec z vrinjenim stavkom v trenutku napade podatkovno bazo (peta vrstica v izseku kode 5.1). Ker sistem v stavek SQL vstavi celotno vsebino vnosnega polja, pride do vrinjenega stavka (osma vrstica v izseku kode 5.1). Celoten stavek SQL tako sestavljajo tri ločene poizvedbe. Prva poizvedba izpiše vse uporabnike, kjer je ime enako črki a. Druga poizvedba pobriše vse podatke iz tabele uporabnikov, tretja poizvedba pa izpiše vse informacije o uporabnikih. Tako je napadalec brez večjega napora prišel do zaupnih podatkov, hkrati pa z izbrisom podatkovne tabele onemogočil pravilno delovanje aplikacije.

Da bi se pred takšnim poskusom napada lahko obvarovali, je bilo v sistemu potrebno implementirati t.i. pripravljene stavke SQL (angl. *Prepared statement*). Od klasičnih stavkov SQL se razlikujejo predvsem v obliki zapisa

ter pripenjanju posameznih parametrov.

Izvajanje pripravljenih stavkov SQL bi lahko v grobem opisali v treh korakih:

1. V prvem koraku se podatkovni bazi pošlje zgolj predloga stavka. Posamezne vrednosti v stavku ostanejo nedefinirane. Tej vrednosti se reče parameter, označimo pa jo z vprašajem (?).

2. Ko podatkovna baza parametriziran stavek prejme, ga prebere, razčleni, prevede in nato optimizira. Rezultat shrani, ne da bi ga dejansko izvedla.

3. V zadnjem koraku aplikacija parametrom pripne dejanske vrednosti, podatkovna baza pa nato stavek izvede in vrne rezultate.

Za lažje razumevanje delovanja pripravljenih stavkov SQL poglejmo še primer v izseku kode 5.2.

```
1  <?php
2  function odstraniStudenta($vpisnaSt, $solLeto){
3
4     $mysqli=connect_database();
5
6     $stmt = $mysqli->prepare('DELETE FROM izmenjave WHERE
          vpisnaSt = ? AND solLeto = ?');
7     $stmt->bind_param('ss', $vpisnaSt, $solLeto);
8     $ress = $stmt->execute();
9
10    $stmt->close();
11    $mysqli->close();
12 }
13 ?>
```

Izsek kode 5.2: Primer uporabe pripravljenih stavkov SQL.

Imamo funkcijo, ki na podlagi vpisne številke in šolskega leta izbriše ustreznega študenta. Stavek SQL sestavimo kot klasičen stavek, le da mesta, kjer bi sicer bile vpisane spremenljivke, nadomestimo z vprašaji. V tem trenutku še ni pomembno, kakšnega tipa bodo vstavljene vrednosti. Ko je stavek uspešno parametriziran, mu moramo določiti vrednosti posameznih parametrov. To storimo tako, da najprej za vse parametre določimo oznako za tip. V našem primeru imamo dva parametra, ki sta tekstovnega tipa, zato kot oznako tipa vpišemo `ss`. Temu sledijo naštete vse spremenljivke, ki bodo v stavku nastopale. Ko so parametri uspešno določeni, lahko stavek izvedemo.

Ker se predloga stavka in parametri pošiljajo ločeno, smo z uporabo pripravljenih stavkov SQL zaščiteni pred tovrstnimi napadi in tako so naši podatki varni.

5.2 Napadi XSS

Prav tako kot vrivanje stavkov SQL je tudi napad XSS (angl. *Cross-site Scripting*) eden izmed najbolj pogostih napadov na spletne strani. Nezaščitene spletne strani tako napadalcu dovoljujejo vstavljanje poljubne skriptne kode v vnosne obrazce [13]. Takšni napadi se največkrat uporabljajo za krajo piškotkov ter drugih spletnih prevar.

```
1 // localhost/index.php?q=<script>alert('Napad!');</script>
2
3 <?php
4 echo $_GET['q'];
5 ?>
```

Izsek kode 5.3: Primer napada XSS.

Za lažje razumevanje poteka napada poglejmo poenostavljen primer v izseku kode 5.3. Predpostavimo, da obstaja preprosta spletna stran, ki prebere vnešen parameter `q` in ga izpiše na zaslon. Ker za varnost nismo poskrbeli,

lahko napadalec kot parameter vnese poljubno besedilo. Tako lahko napadalec kot parameter vnese tudi skriptno kodo (vrstica 1), sistem pa tega ne zazna in jo poskuša izpisati (vrstica 4). Spletna stran izpiše celotno vsebino parametra `q`, kar povzroči izpis opozorila z besedilom '`Napad!`'.

Poskus napada XSS s primera vsebuje zgolj opozorilo, zato tukaj podatki niso ogroženi. A napadalec lahko z malo več napora sestavi obsežnejšo skriptno kodo, s čimer lahko resno ogrozi varnost podatkov.

Takšne poskuse napada v veliki večini prestrežejo že brskalniki, ki ob dvomljivih vnosih posumijo, da bi morda lahko šlo za poskus napada. Ko brskalnik poskus napada zazna, samodejno pretvori posebne znake, s čimer spletni strani pove, da gre za niz posebnih znakov in ne dejansko kodo. Nekateri brskalniki so zasnovani celo tako, da o morebitnem poskusu napada XSS tudi obvestijo uporabnika.

Ker pa lahko posamezni brskalniki pod določenimi pogoji pri preverjanju vnosov tudi odpovejo oz. svojega dela ne opravijo po pričakovanjih, je bilo potrebno v spletni aplikaciji implementirati dodatni mehanizem preverjanja poskusa napada. Tako smo morali na vseh mestih, kjer ima uporabnik možnost vplivati na delovanje aplikacije, zagotoviti, da uporabnik vnese zgolj tisto, kar sistem od njega pričakuje.

Pred tovrstnimi napadi se najučinkoviteje zaščitimo s filtriranjem vnosnih parametrov. Pri tem smo si največ pomagali z ukazoma `htmlspecialchars()` in `strip_tags()`. Prvi ukaz omogoča pretvorbo posebnih znakov v posamezne entitete HTML, primerne za izpis na zaslon. Drugi ukaz pa vnešenemu parametru odstrani vse nedovoljene značke HTML, s čimer onemogočimo poskus vnašanja skriptne kode.

5.3 Ostali varnostni pristopi

Poleg pristopov za preprečevanje omenjenih napadov je bilo potrebno aplikacijo zavarovati tudi pred ostalimi poskusi napada. Tako smo za avtentikacijo prijavljenih študentov uporabili imeniški strežnik, saj so tako občutljivi po-

datki ločeni od same aplikacije (aplikacija prijavnih podatkov študentov nikoli ne shranjuje). Podatki o administratorjih se sicer shranjujejo v klasično podatkovno bazo, a smo tudi tam s kodiranjem gesel poskrbeli za višjo stopnjo varnosti.

Pomemben dejavnik pri zagotavljanju varnosti aplikacije je tudi omejevanje dostopa do posameznih strani. Pri vsaki posamezni podstrani aplikacije smo preverili, ali je uporabnik sploh prijavljen. Če prijave ni, se uporabnika preusmeri na vstopno stran. V primeru, da prijava obstaja, pa smo preverili vrsto prijave in temu primerno prilagodili prikaz.

Tudi pri vnašanju vnosnih parametrov v naslov URL smo definirali možne vnose. V primeru vnosa neveljavnega parametra aplikacija to zazna in takšen parameter ignorira.

S temi pristopi smo aplikaciji zagotovili zadovoljivo stopnjo varnosti in tako poskrbeli, da lahko do aplikacije dostopajo zgolj tisti uporabniki, katerim je aplikacija tudi namenjena.

Poglavje 6

Končne funkcionalnosti

Končne funkcionalnosti spletne aplikacije smo za lažjo razlago razdelili na tri sklope. Prvi sklop so funkcionalnosti, ki jih imata tako študent, kot administrator. Drugi sklop so funkcionalnosti, ki so namenjene študentom, tretji sklop pa so funkcionalnosti, ki jih uporabljajo administratorji.

Funkcionalnosti, ki so skupne obema vrstama uporabnikov, so povezane predvsem s prijavo v aplikacijo. Ob vstopu v aplikacijo se uporabniku prikaže prijavni obrazec, ki zahteva vnos uporabniškega imena in gesla.

Na podlagi vnešenih podatkov sistem preveri, ali gre za študenta ali administratorja ter temu primerno prikaže vsebino. Ko je uporabnik prijavljen v spletno aplikacijo, ima tudi možnost odjave, s čimer se pobrišejo vsi trenutno neshranjeni podatki.

6.1 Funkcionalnosti študenta

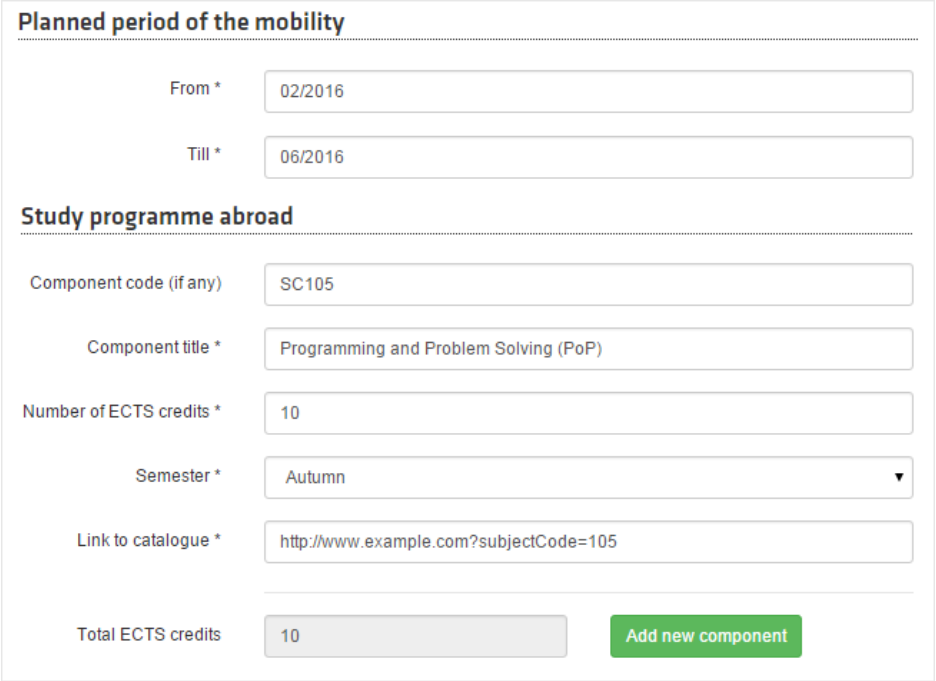
Glavna ter za študenta najbolj pomembna funkcionalnost aplikacije je izpolnjevanje študijskega sporazuma. Ob prijavi v aplikacijo sistem preveri, v kakšnem stanju je trenutno njegov študijski sporazum. V primeru, da študijskega sporazuma študent še ni izpolnil, ga sistem o tem obvesti.

Ob pričetku izpolnjevanja študijskega sporazuma se študentu prikažejo vnosna polja, ki študentu jasno povejo, kaj je potrebno kam vpisati. Ne-

kateri podatki o študentu so že vnaprej znani, zato sistem takšne podatke samodejno vpiše v obrazec. Celoten postopek izpolnjevanja je zaradi obsega razdeljen na več delov.

V prvem delu študent izpolni svoje osebne podatke ter podatke o domači in tuji instituciji. Tukaj tudi izbere odgovorni osebi na obeh institucijah ter določi nivo znanja jezika, v katerem potekajo predavanja na tuji instituciji.

V drugem delu študent s pomočjo vgrajenega koledarja izbere dogovorjeni termin izmenjave ter določi predmetnik, ki ga želi poslušati na gostujoči instituciji (slika 6.1). Študent tukaj tudi izbere, katere predmete na domači instituciji želi imeti priznane.



Slika 6.1: Primer izbora predmeta na tuji instituciji.

V tretjem delu ima študent izpisan celoten pregled izpolnjenega obrazca. Če je obrazec pravilno izpolnjen, ga študent lahko odda. V nasprotnem primeru se lahko vrne na del obrazca, kjer želi opraviti spremembe. Študent ima prav tako vedno na voljo možnost shranjevanja trenutnega obrazca, s

čimer študentu omogočimo kasnejše nadaljevanje izpolnjevanja.

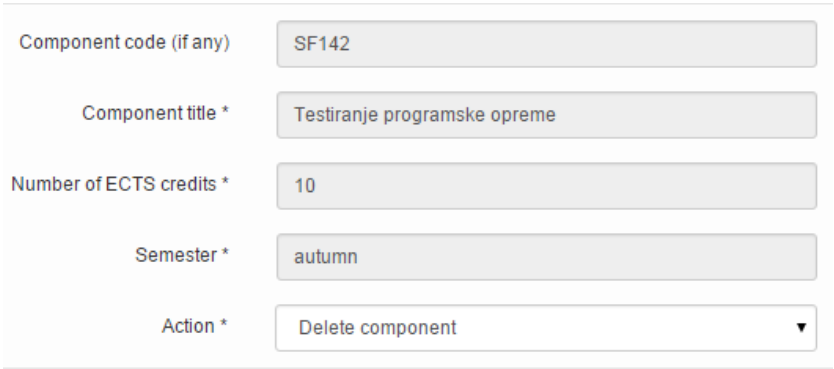
Po oddaji obrazca je potrebno počakati na pregled administratorja. V primeru, da je potreben popravek obrazca, sistem študenta ustrezno obvesti. Študent mora obrazec ponovno izpolniti, pri tem pa mu sistem prikaže, kje je potreben popravek.

Ko je študijski sporazum uspešno izpolnjen ter odobren, lahko študent tudi uveljavlja naknadne spremembe. Tudi obrazec za naknadne spremembe je zaradi boljše preglednosti razdeljen na več delov.

V prvem delu študent označi spremembe, ki bi jih rad uveljavljal na tuji instituciji. Sem spada dodajanje novih in odstranjevanje že obstoječih predmetov ter sprememba termina izmenjave.

V drugem delu študent označi spremembe, ki bi jih rad uveljavljal na domači instituciji (slika 6.2). Tukaj študent tudi označi morebitne spremembe pri odgovorni osebi na tuji instituciji.

Tretji del je namenjen pregledu celotnega obrazca. Tudi tukaj ima študent na voljo shranjevanje in popravljanje obrazca ter oddajo v pregled.



Slika 6.2: Primer zahtevka za spremembo predmeta.

Podobno kot pri prvem obrazcu je tudi tukaj potrebna odobritev s strani administratorja. V primeru, da je obrazec napačno izpolnjen, študent prejme ustrezno obvestilo ter ponovno izpolni obrazec. Če je obrazec pravilno izpolnjen, se spremembe uveljavijo tudi v študentovem študijskem sporazumu.

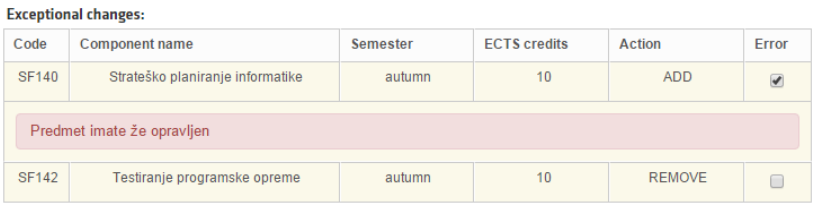
Po vsaki odobritvi obrazcev je potrebno obrazec tudi natisniti in pridobiti ustrezne podpise (študijski sporazum morajo podpisati študent, koordinator na domači instituciji in koordinator na tuji instituciji). Študent ima v vsakem trenutku dostop do podstrani, ki prikaže izpolnjen obrazec v formatu, primernem za tisk. Tako lahko študent obrazec natisne ali pa ga shrani na svoj računalnik.

V primeru morebitnih nejasnosti ali težav ima študent na voljo tudi tehnično pomoč, kjer izpolni obrazec ter ga pošlje odgovornemu administratorju.

6.2 Funkcionalnosti administratorja

V primerjavi s funkcionalnostmi študenta ima administrator aplikacije bistveno več funkcionalnosti. Ob pričetku uporabe aplikacije lahko administrator v sistem uvozi zbirko študentov, ki bodo odšli na študijsko izmenjavo. Administrator lahko po želji zbirko študentov ter njihovih podatkov tudi izvozi za kasnejšo obdelavo.

Poleg tega ima administrator na voljo tudi dve tabeli, ki prikazujeta seznam vseh študentov. V prvi tabeli so izpisani vsi študentje, ki imajo dodeljen dostop do aplikacije. Tukaj je prikazan tudi status, v katerem se trenutno njihov študijski sporazum nahaja ter možne akcije. V drugi tabeli so izpisani tisti študentje, ki so podali zahtevo po uveljavljanju naknadnih sprememb.



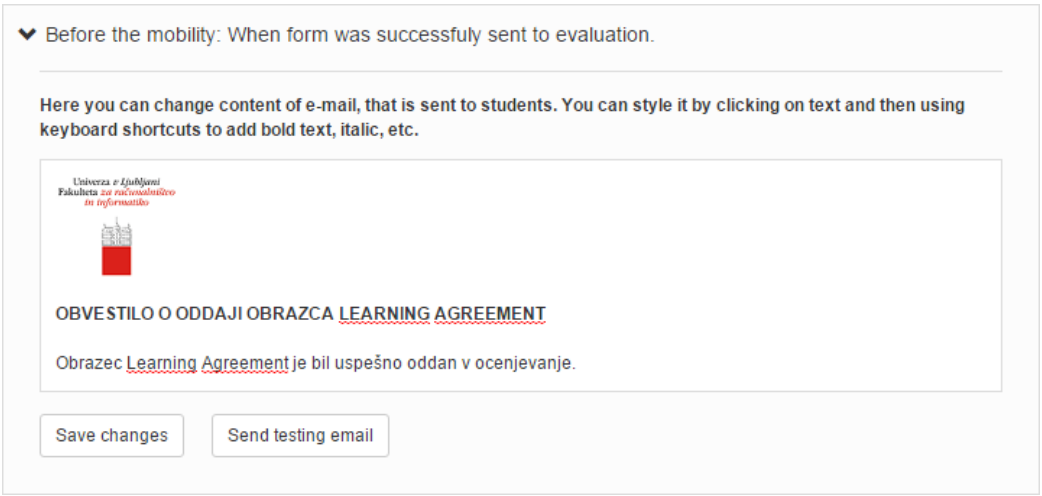
Slika 6.3: Primer pregledovanja obrazcev.

Ko študent posamezen obrazec pošlje v pregled, aplikacija to administra-

torju prikaže v ustrezni tabeli. Pregled obrazcev poteka tako, da administrator v posameznem sklopu označi ali je vnos pravilen ali ne ter po potrebi pripiše komentar (slika 6.3).

Poleg funkcionalnosti, ki zadevajo pregled in potrjevanje študijskih sporazumov, ima administrator na voljo tudi ostale funkcionalnosti. Tako lahko administrator tudi ureja osnovne podatke o domači instituciji, kot so sedež fakultete ter kontaktni podatki odgovorne osebe za študijske izmenjave.

Administrator lahko prav tako dodaja nove oz. briše že obstoječe administratorje. Pri dodajanju novih administratorjev lahko določi tudi skupino, ki ji pripada novi administrator. Na voljo sta skupini za tehnično pomoč ter za prejemanje sporočil. Skupina za tehnično pomoč združuje tiste administratorje, ki bodo prejemali elektronska sporočila o napakah in prošnjah za pomoč. Skupina za prejemanje sporočil pa združuje tiste administratorje, ki bodo prejemali obvestila o poteku izpolnjevanja obrazcev posameznih študentov.



Slika 6.4: Primer urejanja vsebine sporočil.

Poleg ostalih funkcionalnosti lahko administrator tudi ureja vsebino sporočil, ki se študentu pošiljajo ob posameznih akcijah. Tako lahko na primer administrator določi, kakšno sporočilo se pošlje študentu ob uspešno izpolnjenem

obrazcu.

Kot je razvidno s slike 6.4, lahko administrator vsebino ureja v naprednem urejevalniku, vgrajenem v spletni aplikaciji. Tako lahko na enostaven način med vsebino dodaja tudi slike ter ostale stilske podobe. Urejevalnik omogoča tudi spreminjanje oblike pisave. S kombinacijami tipk `Ctrl+B`, `Ctrl+I` ter `Ctrl+U` tako lahko hitro pisavo odebelimo, jo podčrtamo ali pišemo poševno. Urejevalnik omogoča tudi enostavno dodajanje slik v sporočila. To storimo tako, da na spletu poiščemo sliko željenih dimenzij, jo kopiramo v odložišče, nato pa prilepimo v urejevalnik. Aplikacija povezavo do slike shrani v podatkovno bazo, s čimer zagotovi, da bodo vsi prejemniki sporočila lahko videli to sliko.

Poglavje 7

Testiranje

Pred začetkom javne uporabe spletne aplikacije je bilo potrebno izvesti še zaključno testiranje, s čimer smo se želeli prepričati, da aplikacija deluje na vseh napravah ter v vseh brskalnikih.

Testiranje je sicer potekalo že med samim razvojem, zato je zaključno testiranje potekalo hitreje, kot bi sicer. V nadaljevanju so predstavljeni štirje ključni sklopi testiranja spletne aplikacije.

7.1 Testiranje v različnih brskalnikih

Ker trenutno na trgu obstaja veliko število različnih brskalnikov, je bilo potrebno zagotoviti, da aplikacija v vseh brskalnikih teče nemoteno.

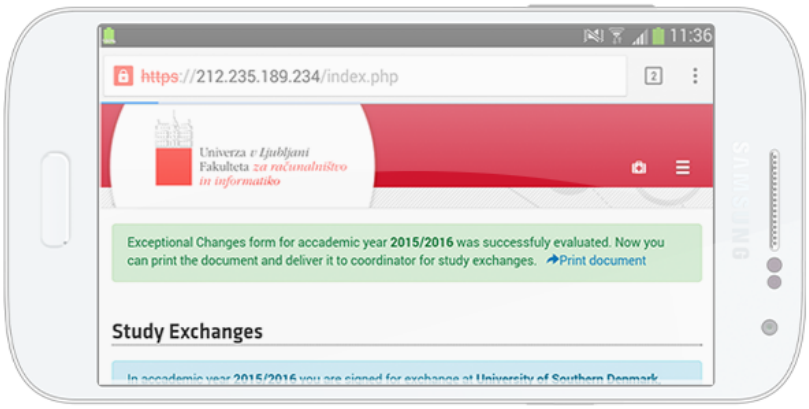
Prikaz spletne strani se v različnih brskalnikih največkrat razlikuje v grafični podobi, kjer pridejo določeni elementi postavljeni drugače, kot smo načrtovali. V ta namen je zato potrebno na spletni strani preverjati, kakšen brskalnik uporabljamo, in temu primerno elemente tudi razporediti.

Ker izgled same aplikacije vizualno ni tako zahteven, večjih popravkov ni bilo potrebnih. Morebitne napačne prikaze je reševala že knjižnica Bootstrap, ki smo jo uporabljali za osnovno ogrodje spletne strani. Tako smo stalno zagotavljali enako prikazovanje v vseh brskalnikih.

7.2 Testiranje na različnih napravah

Tehnologija je v zadnjih nekaj letih zelo napredovala in tako imajo študentje sedaj na voljo tudi pametne telefone ter tablice, s katerih lahko v vsakem trenutku dostopajo do spletnih vsebin. Ker smo tudi mi želeli razviti spletno aplikacijo, ki bo dostopna z vseh naprav, je bilo potrebno s testiranjem to tudi zagotoviti.

Prikaz in delovanje spletne aplikacije smo testirali tako na osebnih računalnikih, kot tudi prenosnih računalnikih. Tako smo želeli zagotoviti, da se aplikacija prikazuje pravilno tudi pri različnih velikostih zaslona.



Slika 7.1: Ležeči pogled aplikacije na pametnem telefonu.

Prav tako smo izvedli testiranje na pametnih telefonih ter tablicah. V pomoč nam je bila tudi tukaj že večkrat omenjena knjižnica Bootstrap, ki je poskrbela za pravilno postavljanje elementov na spletno stran.

Ker imajo pametni telefoni in tablice implementirana dva različna pogleda na zaslon (pokončen in ležeč), je bilo potrebno preveriti tudi način prikaza spletne aplikacije v posameznem pogledu (slika 7.1).

7.3 Testiranje varnosti

V sklopu testiranja varnosti je bilo potrebno zagotoviti, da je spletna aplikacija dovolj dobro zaščitena pred posameznimi poskusi napada.

Za preverjanje varnosti smo pripravili nekaj testnih razredov (izsek kode 7.1), ki bi ob dobro zaščiteni aplikaciji morali biti zaznani kot poskus napada. V posamezna vnosna polja smo tako skušali vriniti zlonamerno kodo, ki bi porušila normalno delovanje aplikacije.

```
1 Name: <script> alert('napad'); </script>
2 Function: <script> alert('napad'); </script>
3 Phone number: <script> alert('napad'); </script>
4 E-mail: <script> alert('napad'); </script>
```

Izsek kode 7.1: Primer dela testnega razreda.

Ko je aplikacija poskusne napade uspešno prestala, smo lahko začeli s testiranjem omejevanja dostopa. Tukaj smo testirali, ali je dostop do posamezne datoteke pravilno pogojen (npr. prijavljen uporabnik nima dovoljenja za dostop do podstrani za prijavo).

V tem sklopu testiranja smo preverili tudi omejevanje dostopa do konfiguracijskih datotek, saj smo želeli zagotoviti, da obiskovalec spletne aplikacije v nobenem trenutku ne mora dostopati do datotek, ki določajo zgradbo in delovanje spletne aplikacije ali pa vsebujejo občutljive podatke.

7.4 Končno testiranje

V razvojnem okolju smo za izdelavo spletne aplikacije uporabljali strežniški paket XAMPP, ki je deloval na operacijskemu sistemu Windows. Izvajalno okolje, v katerem je aplikacija nameščena in dostopna študentom, pa temelji na operacijskem sistemu Linux. Zato je bilo potrebno po namestitvi aplikacije v izvajalno okolje ponovno testirati vse pomembnejše sklope.

Posebno pozornost je bilo potrebno nameniti funkcionalnostim aplikacije, ki bi lahko zaradi drugačnega operacijskega sistema prenehale delovati ali pa ne bi delovale v skladu s pričakovanji.

Tako je bilo potrebno preveriti pošiljanje sporočil, ustvarjanje datotek PDF ter uvoz in izvoz podatkov o študentih.

Prav tako je bilo potrebno v izvajalnem okolju popraviti nekatere konfiguracije sistema, da se lahko aplikacija nemoteno izvaja. Omogočiti je bilo potrebno uporabo predpripravljenih stavkov za zaščito pred vrivanjem stavkov SQL ter omogočiti uporabo t.i. skrajšanih značk (angl. *Short tags*).

Poglavje 8

Sklepne ugotovitve

Razvoj spletne aplikacije smo uspešno zaključili, saj smo realizirali vse naloge, ki smo si jih na začetku zadali. Končni izdelek je delujoča spletna aplikacija, ki študentom omogoča izpolnjevanje študijskih sporazumov, administratorjem pa pregled in potrjevanje le-teh. Poleg tega lahko administratorji tudi urejajo vsebino elektronskih sporočil, ki se pošiljajo študentom ob zaključku posameznih akcij. Poskrbeli smo tudi za zadovoljivo stopnjo varnosti, s čimer smo aplikacijo zaščitili pred morebitnimi poskusi napada. Strukturo spletne aplikacije smo prilagodili tako, da omogoča pravilen prikaz vsebine tudi na različnih napravah. Tako lahko študentje študijske sporazume izpolnjujejo ali popravljajo tudi na pametnih telefonih ter tablicah.

Razvoj aplikacije je zaradi jasnih in dobro definiranih zahtev potekal brez večjih problemov, saj smo v vsakem trenutku vedeli, kakšen mora biti naslednji korak.

Aplikacija bo študentom in odgovornim osebam za študentske izmenjave poenostavila potek izpolnjevanja študijskih sporazumov, saj omogoča enostavno usklajevanje vsebine študijskega sporazuma. Tako lahko študent prinese koordinatorju v podpis le predhodno usklajen in potrjen obrazec, kar odpravi dolgotrajna preverjanja in usklajevanja vsebine obrazca na govorilnih urah pri koordinatorju. Tako je tudi samo podpisovanje veliko hitrejše, saj koordinatorju ni potrebno preverjati še vsebine študijskega sporazuma.

Aplikacija omogoča tudi izvoz podatkov o študentih v formatih `.xlsx` in `.csv` in tako administratorjem omogoča nadaljno obdelavo pridobljenih podatkov. Tudi sicer aplikacija ponuja možnost nadaljnega razvoja in razširitve. Besedilo, ki se prikazuje v aplikaciji, je shranjeno ločeno od same izvorne kode, zato bi lahko v aplikaciji kasneje enostavno implementirali tudi podporo večjezičnosti, če bi bilo potrebno.

Ker obstaja možnost, da se bo struktura obrazcev za študijski sporazum v prihodnosti še spreminjala, smo ustvarili ločeno stilsko predlogo, ki določa postavitev glavnih elementov dokumenta. Obrazci se v dokument PDF preslikajo s pomočjo tabel HTML, zato lahko z nekaj programerskega znanja strukturo obrazcev tudi dodatno prilagodimo.

Literatura

[1] J. Lockhart, "Modern PHP: New features and good practices", O'Reilly Media, 2015.

[2] D. Sklar, A. Trachtenberg, "PHP Cookbook: Solutions & examples for PFP programmers", O'Reilly Media, 2014.

[3] J. Spurlock, "Bootstrap: Responsive Web Development", O'Reilly Media, 2013.

[4] 10 Usability Heuristics for User Interface Design. [Online]. Dosegljivo: https://www.nngroup.com/articles/ten-usability-heuristics/. [Dostopano 20. 09. 2015].

[5] 5 Helpful Tips for Creating Secure PHP Applications. [Online]. Dosegljivo: http://code.tutsplus.com/tutorials/5-helpful-tips-for-creating-secure-php-applications--net-2260. [Dostopano 20. 09. 2015].

[6] Cascading Style Sheets. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano 20. 09. 2015].

[7] HTML. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/HTML. [Dostopano 20. 09. 2015].

[8] Kaj je Front End Developer?. [Online]. Dosegljivo: http://gregag.com/blog/kaj-je-front-end-developer. [Dostopano 22. 10. 2015].

[9] LDAP. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/LDAP. [Dostopano 22. 10. 2015].

[10] Oblikovni trendi spletne strani za leto 2015/2016. [Online]. Dosegljivo: http://linea-studio.com/index.php/2015/10/26/spletne_strani-oblikovni_trendi_2015_2016/. [Dostopano 20. 09. 2015].

[11] PHP. [Online]. Dosegljivo: https://sl.wikipedia.org/wiki/PHP. [Dostopano 18. 01. 2016].

[12] SQL. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/SQL. [Dostopano 20. 09. 2015].

[13] XSS Attack Examples. [Online]. Dosegljivo: http://www.thegeekstuff.com/2012/02/xss-attack-examples/. [Dostopano 20. 09. 2015].